METR4202 Tutorial 8 Solutions

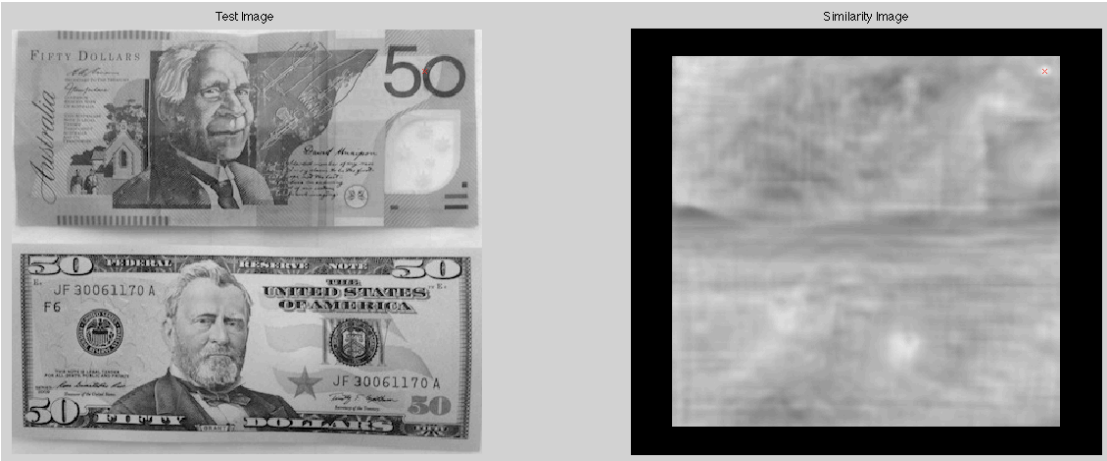
1) -

2) To robustly describe a $50 note, you need to talk about its appearance, brightness, colour etc. These are the kind of things that image features attempt to encode.

The following code can be used for part 2.

```
% Read images
template = rgb2gray(imread('fifty_text.jpg'));
test_1 = rgb2gray(imread('test_1.jpg'));

% Find correlations using Zero Normalised Cross Correlation
% NB: The last parameter is unneccessary
% This step might take a while...
S = isimilarity(template, test_1, @zncc);

% Find the index of the best match
indices = find(S == max(max(S)));

% Convert the (1D) index to (2D) matrix subscript values
[y x] = ind2sub(size(S), indices);

% Plot the two images
figure;
subplot(1,2,1);
imshow(test_1);
hold on;
plot(x, y, 'rx');
title('Test Image');

subplot(1,2,2);
imshow(S);
hold on;
plot(x, y, 'rx');
title('Similarity Image');
```

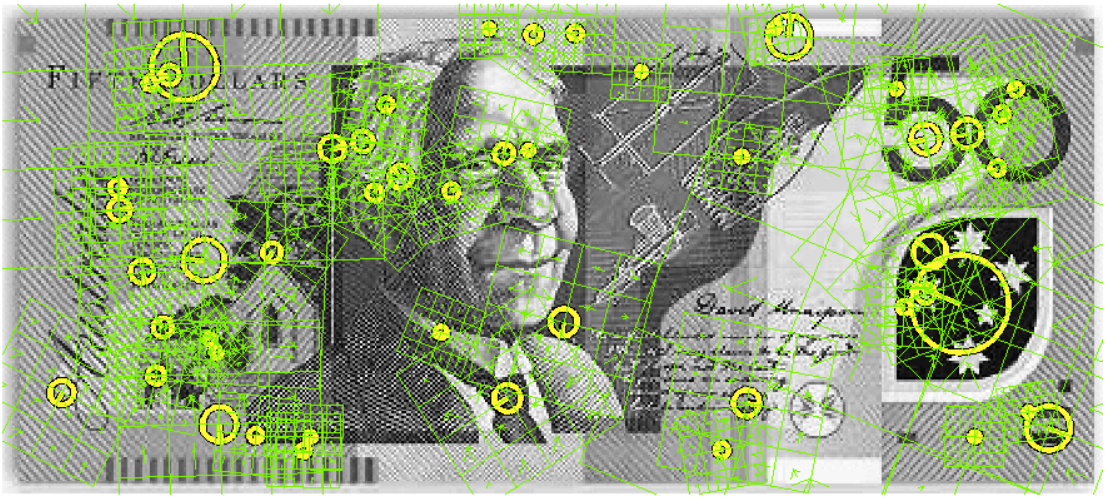
The output produced by the above code. The best matching pixel is marked (look closely) on both images.



3) The following code can be used for part 3.

```
% Read training images
train_back = rgb2gray(imread('training_back.jpg'));
train_front = rgb2gray(imread('training_front.jpg'));

% Extract SIFT features and descriptors
[f_train_front d_train_front] = vl_sift(single(train_front));
[f_train_back d_train_back] = vl_sift(single(train_back));

% Show features and descriptors for front image
figure;
imshow(train_front);

perm = randperm(size(f_train_front, 2));
sel = perm(1:50);

h1 = vl_plotframe(f_train_front(:, sel));
h2 = vl_plotframe(f_train_front(:, sel));
set(h1, 'color', 'k', 'linewidth', 3);
set(h2, 'color', 'y', 'linewidth', 2);

h3 = vl_plotsiftdescriptor(d_train_front(:, sel), f_train_front(:, sel));
set(h3, 'color', 'g');
```

The output image;

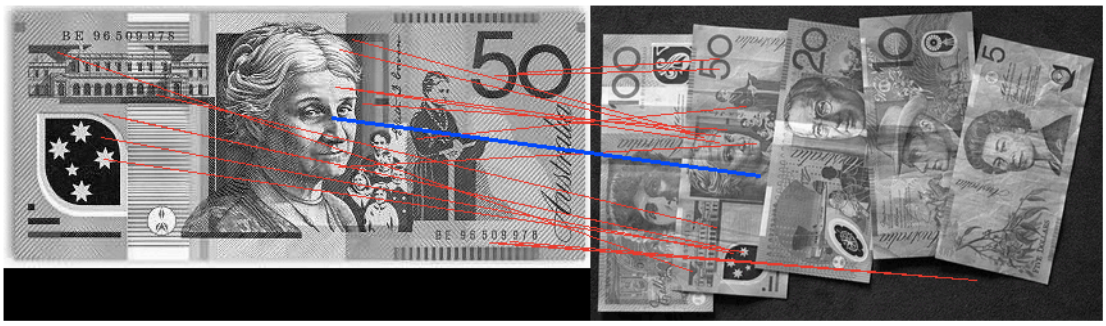


4) The following code can be used to solve part of question 4.

```
% Read testing images
test_1 = rgb2gray(imread('test_1.jpg'));
test_2 = rgb2gray(imread('test_2.jpg'));

% Extract features and descriptors
[f_test_1 d_test_1] = vl_sift(single(test_1));
[f_test_2 d_test_2] = vl_sift(single(test_2));

% Match the second test image
% Note that we had to specify that we wanted to use the back training image
% database (prior knowledge was required)
[matches scores] = vl_ubcmatch(d_train_back, d_test_2);

% Sort the scores
[scores_sorted, indices] = sort(scores, 'descend');

% Extract the top 10 matches
```

```
matches = matches(:, indices(1:10));

% Visualise the matches
[input_points base_points] = visualise_sift_matches(train_back, test_2,
f_train_back, f_test_2, matches);
```

The output image;



Choosing the best N feature matches assumes that there were some good matches, which implicitly assumes the object was present in the test scene. If this was not the case, such a method will likely give false positives. In contrast, a threshold will prevent the above from happening, however assumes that object features in two different scenes will have similar feature scores, which may not always be the case.

Feature based methods generally fail when the image lacks ‘information density’. This is a hard parameter to quantify, however can be roughly correlated to image contrast. For example running SIFT to try and detect a plain object (such as a silver coin) will likely fail, as there is not much contrast in the training image, and hence no good features can be found. Image Features also generally require higher resolutions than ‘dumb’ classifiers (e.g. color) in order to be useful.